Lampiran

Lampiran 1. Kode Program

A. Program Forklift Roda 2

Main_Program.ino

```
//Terms of use
//THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
//IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
//FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL THE
//AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER
//LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM,
//OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN
//THE SOFTWARE.
////////////////////////////////////////////////////////////////////
//////////////////
//looping data mpu setelah 1 cycle step roda
#include "Joystick.h"
#include "MotorStep.h"
#include "PID.h"
#include "SerialMon.h"
#include "mpu6050sudut.h"
#include "Strains.h"
#include "Arduino.h"
#include "HX711.h"
#include <Wire.h>
#include <RF24.h>
#include <RF24Network.h>
#include <SPI.h>

RF24 radio(9,10);
RF24Network network(radio);
```

```
byte start;
int jumlah_sample=500;
int counteri;
int left_motor, throttle_left_motor, throttle_counter_left_motor,
throttle_left_motor_memory;
int right_motor, throttle_right_motor,
throttle_counter_right_motor, throttle_right_motor_memory;
long kal_aPitch,kal_aRoll;
byte data;

  unsigned long intervaltotal=1000;
  unsigned long waktutotalsebelum=0;
  unsigned long wt;
  unsigned long it;
  unsigned long bt;
  int berat_counter;
  int berat;
  const uint16_t this_node = 00;// Master
  const uint16_t node01 = 01;// Slave 1 aka remote controller
  const uint16_t node02 = 02;// Slave 2 aka pengolah beban
  int incomingData2;
  int incomingData;
  int incomingdata;
  int data1;
  int data2;

void setup()
{
  Serial.begin(9600);      // Baud rate Serial Monitor
  SPI.begin();
  radio.begin();
  network.begin(90,this_node);
  radio.setDataRate(RF24_2MBPS);
  //massa.begin(38400);
  //Bluetooth.begin(38400); // Baud rate Transmisi Bluetooth
untuk transfer data
  //Bluetooth1.begin(19200); // Baud rate Transmisi Bluetooth
untuk Kontrol Motor dengan Joystick
  Serial.println("Bluetooth Terdeteksi");
  Wire.begin(); // Mengaktifkan Library Wire.h
  TWBR =12;   //Set the I2C clock speed to 400kHz
  //To create a variable pulse for controlling the stepper motors
a timer is created that will execute a piece of code (subroutine)
every 20us
```

```
//This subroutine is called TIMER2_COMPA_vect
  TCCR2A = 0;                //Make sure that the TCCR2A register
is set to zero
  TCCR2B = 0;                //Make sure that the TCCR2A register
is set to zero
  TIMSK2 |= (1 << OCIE2A);   //Set the interupt enable bit OCIE2A
in the TIMSK2 register
  TCCR2B |= (1 << CS21);     //Set the CS21 bit in the TCCRB
register to set the prescaler to 8
  OCR2A = 39;                //The compare register is set to 39
=> 20us / (1s / (16.000.000MHz / 8)) - 1
  TCCR2A |= (1 << WGM21);    //Set counter 2 to CTC (clear timer
on compare) mode

  /*======Power Management======*/
  Wire.beginTransmission(0x68);//Power Management
  Wire.write(0x6B);
  Wire.write(0b00001000);
  Wire.endTransmission();

  /*======Register Konfigurasi Accelorometer======*/
  Wire.beginTransmission(0x68); //Memulai transmisi pada alamat
MPU6050
  Wire.write(0x1C);       // Mengaktifkan alamat Register
Konfigurasi Accelorometer
  Wire.write(0b00000111); // Setting bit-bit pada alamat tersebut
AFS_SEL bit 4 = 0,bit 3 = 0,Full Scale Range +- 2g
  Wire.endTransmission(); // Mengakhiri Transmisi

  /*======Register Konfigurasi Gyroscope======*/
  Wire.beginTransmission(0x68);//Memulai transmisi pada alamat
MPU6050
  Wire.write(0x1B);      // Mengaktifkan alamat Register
Konfigurasi Gyroscope
  Wire.write(0b00000000);// Setting bit-bit pada alamat tersebut
AFS_SEL bit 4 = 0,bit 3 = 0,Full Scale Range +- 250 derajat/detik
  Wire.endTransmission();// Mengakhiri Transmisi

  /*======Register Konfigurasi DLPF======*/
  Wire.beginTransmission(0x68);//Memulai transmisi pada alamat
MPU6050
  Wire.write(0x1A);      // Mengaktifkan alamat Register
Konfigurasi DLPF (Digital Low Pass Filter)
```

```
  Wire.write(0b00000011);// Setting bit-bit pada alamat tersebut
bit
  Wire.endTransmission();// Mengakhiri Transmisi

// Konfigurasi PIN
    pinMode(Step_1, OUTPUT);
    pinMode(Step_2, OUTPUT);
    pinMode(Dir_1, OUTPUT);
    pinMode(Dir_2, OUTPUT);

  /*=============Kalibrasi Gyroscope==============*/
  for (int i=0; i < 500; i++)
  {
  Wire.beginTransmission(0x68);
     //Start communicating with the MPU-6050
  Wire.write(0x43);
     //Send the requested starting register
  Wire.endTransmission();
     //End the transmission
  Wire.requestFrom(0x68,4);
    //Request 14 bytes from the MPU-6050
  while(Wire.available() <
4);                                          //Wait until all the
bytes are received

  gyro_x_cal_value +=
Wire.read()<<8|Wire.read();                                  //Add
the low and high byte to the gyro_x variable
  gyro_y_cal_value +=
Wire.read()<<8|Wire.read();                                  //Add
the low and high byte to the gyro_y variable

  delayMicroseconds(3000);
  }

  gyro_x_cal_value /= 500;
  gyro_y_cal_value /= 500;
  Serial.print("Kalibration Gyro X=");
  Serial.print(gyro_x_cal_value);
  Serial.print("Kalibration Gyro Y=");
  Serial.println(gyro_y_cal_value);
wt=micros()+4000;

}
```

```
void loop()

{
  mpu6050();
  strain();
  gerak_pid();
  Joys();
  jumlahstep();
  Serial.print(aPitch);
  Serial.print(',');
  Serial.print(angle_gyroraw);
  Serial.print(',');
  Serial.println(angle_gyro);
  while(wt > micros());
  wt += 4000;
}
ISR(TIMER2_COMPA_vect){
  //Left motor pulse calculations
  throttle_counter_left_motor ++; //Increase the
throttle_counter_left_motor variable by 1 every time this routine
is executed
  if(throttle_counter_left_motor > throttle_left_motor_memory){
//If the number of loops is larger then the
throttle_left_motor_memory variable
    throttle_counter_left_motor = 0; //Reset the
throttle_counter_left_motor variable
    throttle_left_motor_memory = throttle_left_motor;  //Load the
next throttle_left_motor variable
    if(throttle_left_motor_memory < 0){  //If the
throttle_left_motor_memory is negative
      PORTD &= 0b01111111;  //Set output 3 low to reverse the
direction of the stepper controller
      throttle_left_motor_memory *= -1;  //Invert the
throttle_left_motor_memory variable
    }
    else PORTD |= 0b10000000; //Set output 3 high for a forward
direction of the stepper motor
  }
  else if(throttle_counter_left_motor == 1)PORTD |=
0b01000000;  //Set output 2 high to create a pulse for the
stepper controller
  else if(throttle_counter_left_motor == 2)PORTD &=
0b10111111;  //Set output 2 low because the pulse only has to
last for 20us
```

```
//right motor pulse calculations
  throttle_counter_right_motor ++;  //Increase the
throttle_counter_right_motor variable by 1 every time the routine
is executed
  if(throttle_counter_right_motor >
throttle_right_motor_memory){//If the number of loops is larger
then the throttle_right_motor_memory variable
    throttle_counter_right_motor = 0;  //Reset the
throttle_counter_right_motor variable
    throttle_right_motor_memory = throttle_right_motor;  //Load
the next throttle_right_motor variable
    if(throttle_right_motor_memory < 0){  //If the
throttle_right_motor_memory is negative
      PORTD |= 0b00100000;  //Set output 5 low to reverse the
direction of the stepper controller
      throttle_right_motor_memory *= -1;  //Invert the
throttle_right_motor_memory variable
    }
    else PORTD &= 0b11011111;  //Set output 5 high for a forward
direction of the stepper motor
  }
  else if(throttle_counter_right_motor == 1)PORTD |= 0b00010000;
//Set output 4 high to create a pulse for the stepper controller
  else if(throttle_counter_right_motor == 2)PORTD &= 0b11101111;
//Set output 4 low because the pulse only has to last for 20us
}
```

Joystick.h

```
#ifndef JOYSTICK_H_
#define JOYSTICK_H_

  int turning_speed =30;
  int maxspeed_f =200;
  int maxspeed_b =200;

#endif
```

Joystick.ino

```
#include "Joystick.h"
#include "MotorStep.h"
#include "PID.h"
#include "SerialMon.h"
#include "mpu6050sudut.h"
#include "Strains.h"
#include "Arduino.h"
#include "HX711.h"

void Joys()
{
  //radio.startListening();
  // delay(5);

 network.update();
 radio.startListening();

   while (network.available()>0)
  {
   RF24NetworkHeader header1;

    network.read(header1, &incomingdata, sizeof(incomingdata));

    if(header1.from_node ==1)
    {data1=incomingdata;}

    if(header1.from_node ==2)
    {data2=incomingdata;}

   //Serial.println(data1);  //monitor transfer data master dan
slave 1 (remote controller)
   //Serial.println(data2);  //monitor transfer data master dan
slave 2 (pengolah beban)

  radio.stopListening();
  RF24NetworkHeader header2(01);
  bool ok1 = network.write(header2, &aPitch, sizeof(aPitch));
  }

  if(data_counter < 25)data_counter++;
  else data = 0x00;
```

```
if (data1 == 0x01){//f
    //  error=0;
     // angle_gyro=0;
     if(error<0)error+=0.05;
     else{
if(pid_setpoint <= 1.5 && pid_setpoint >= 0)pid_setpoint +=0.05;
     if(pid_output < maxspeed_f)pid_setpoint += 0.010;
     }
     //Serial.println(pid_setpoint);
     //Serial.println("depan");
    }

if (data1 == 0x01){ // Gerak Maju
    //  error=0;
     // angle_gyro=0;
     if(error<0)error+=0.05;
     else{
if(pid_setpoint <= 1.5 && pid_setpoint >= 0)pid_setpoint +=0.05;
     if(pid_output < maxspeed_f)pid_setpoint += 0.010;
     }
     //Serial.println(pid_setpoint);
     //Serial.println("depan");
    }

else if (data1 == 0x02){// Gerak Mundur
    //error=0;
   // angle_gyro=0;
   if(error>0)error-=0.05;
   else{
      if(pid_setpoint >= - 1.5 && pid_setpoint <= 0)pid_setpoint
-=0.05;
      if(pid_output > maxspeed_b * -1)pid_setpoint -= 0.010;
      //Serial.println(pid_setpoint);}
      //Serial.println("belakang");
    }
    }

else if (data1 == 0x03){ // Belok Kanan
    //if(pid_setpoint <= 0.15)pid_setpoint +=0.01;
    pid_output_left  -= 35;  //Increase the left motor speed
    pid_output_right += 35;
   // Serial.println("kanan");
   }
```

```
else if (data1 == 0x04){//l
    //if(pid_setpoint <= 0.15)pid_setpoint +=0.01;
    pid_output_left  +=
35;                                          //Decrease the left
motor speed
    pid_output_right -= 35;
   //   Serial.println("kiri");
   }

    else if (data1 == 0x00){
    //pid_setpoint=0;
    if(pid_setpoint > 0)pid_setpoint -=0.02;  //If the PID
setpoint is larger then 0.5 reduce the setpoint with 0.05 every
loop
    else if(pid_setpoint < 0)pid_setpoint +=0.02;  //If the PID
setpoint is smaller then -0.5 increase the setpoint with 0.05
every loop
    else pid_setpoint = 0;
 }

}
```

MotorStep.h

```
#ifndef MOTORSTEP_H_
#define MOTORSTEP_H_

#define Step_1 6
#define Step_2 4
#define Dir_1  7
#define Dir_2  5

#endif
```

MotorStep.ino

```
#include "Joystick.h"
#include "MotorStep.h"
#include "PID.h"
#include "SerialMon.h"
#include "mpu6050sudut.h"
#include "Strains.h"
#include "Arduino.h"
#include "HX711.h"
```

```
void jumlahstep()
  {
  //Motor pulse calculations

  //To compensate for the non-linear behaviour of the stepper
motors the folowing calculations are needed to get a linear speed
behaviour.
  if(pid_output_left > 0)pid_output_left = 405 -
(1/(pid_output_left + 9)) * 5500;
  else if(pid_output_left < 0)pid_output_left = -405 -
(1/(pid_output_left - 9)) * 5500;

  if(pid_output_right > 0)pid_output_right = 405 -
(1/(pid_output_right + 9)) * 5500;
  else if(pid_output_right < 0)pid_output_right = -405 -
(1/(pid_output_right - 9)) * 5500;

  //Calculate the needed pulse time for the left and right
stepper motor controllers
  if(pid_output_left > 0)left_motor = 400 - pid_output_left;
  else if(pid_output_left < 0)left_motor = -400 -
pid_output_left;
  else left_motor = 0;

  if(pid_output_right > 0)right_motor = 400 - pid_output_right;
  else if(pid_output_right < 0)right_motor = -400 -
pid_output_right;
  else right_motor = 0;

  //Copy the pulse time to the throttle variables so the
interrupt subroutine can use them
  throttle_left_motor = left_motor;
  throttle_right_motor = right_motor;


  //Loop time timer

  //The angle calculations are tuned for a loop time of 4
milliseconds. To make sure every loop is exactly 4 milliseconds a
wait loop
  //is created by setting the loop_timer variable to +4000
microseconds every loop.
  }
```

PID.h

```
#ifndef PID_H_
#define PID_H_

////Mencari nilai Kcr////

//float Kp=50;
//float Ki=0;
//float Kd=0;

////Nilai PID sebelum fine tuning////

//float Kp=30;
//float Ki=47.5;
//float Kd=4.737;

////Nilai PID setelah fine tuning////

float Kp=46;
float Ki=2.2;
float Kd=12;

float self_balance_pid_setpoint;
float pid_setpoint;

float pid;
float pid_output;
float pid_i_mem;
float pid_output_left, pid_output_right;

#endif
```

PID.ino

```
#include "Joystick.h"
#include "MotorStep.h"
#include "PID.h"
#include "SerialMon.h"
#include "mpu6050sudut.h"
#include "Strains.h"
#include "Arduino.h"
#include "HX711.h"
```

```
void gerak_pid()
 {
    /*=====Stetting PID=====*/
  error =  angle_gyro - pid_setpoint - pid_setmov ;
  //jika ada gangguan luar
  if(pid_output > 10 || pid_output < -10)error += pid_output *
0.012 ;

  pid_i_mem += Ki * error; //Calculate the I-controller value and
add it to the pid_i_mem variable
  if(pid_i_mem > 400)pid_i_mem = 400; //Limit the I-controller to
the maximum controller output
  else if(pid_i_mem < -400)pid_i_mem = -400;

  pid_output = Kp * error + pid_i_mem + Kd * (error -
last_error);

  if (pid_output>180)pid_output=200;
  if (pid_output<-180)pid_output=-200;

  if(aPitch>30 || aPitch <-30)pid_output=0;

  last_error = error;

  if(pid_output < 10 && pid_output > -10)pid_output = 0; //Create
a dead-band/differential gap to stop the motors when the robot is
balanced

  pid_output_left =
pid_output;                                                 //Copy
the controller output to the pid_output_left variable for the
left motor
  pid_output_right =
pid_output;                                               //Copy the
controller output to the pid_output_right variable for the right
motor
  ////////////////////////////////////////////////
 }
```

Strains.h

```
#ifndef STRAINS_H_
#define STRAINS_H_

float pid_setmov; // penetapan set-point berdasarkan beban yang
diangkat
//int range;
const int min_beban = 0;
const int max_beban = 100;

#endif
```

Strains.ino

```
#include "Joystick.h"
#include "MotorStep.h"
#include "PID.h"
#include "SerialMon.h"
#include "mpu6050sudut.h"
#include "Strains.h"
#include "Arduino.h"
#include "HX711.h"

void strain()
{
  int ranges = map(data2, min_beban, max_beban, 0, 9);
  // Serial.print("Reading:");
  // Serial.println(ranges);

switch (ranges)
  {

  case 0:  //range 0-10
    if (pid_setmov <= 0)pid_setmov += 0.025;
    if (pid_setmov > 0)pid_setmov = 0;
break;
  case 1:  //range 10-20
    if (pid_setmov > -1.1)pid_setmov -= 0.025;
    if (pid_setmov < -1.1)pid_setmov += 0.025;
break;
```

```
  case 2:  //range 20-30
    if (pid_setmov > -1.8)pid_setmov -= 0.025;
    if (pid_setmov < -1.8)pid_setmov += 0.025;
break;
  case 3:  //range 30-40
    if (pid_setmov > -2.6)pid_setmov -= 0.025;
    if (pid_setmov < -2.6)pid_setmov += 0.025;
break;
  case 4:  //range 40-50
    if (pid_setmov > -3.3)pid_setmov -= 0.025;
    if (pid_setmov < -3.3)pid_setmov += 0.025;
break;
  case 5:  //range 50-60
    if (pid_setmov > -3.6)pid_setmov -= 0.025;
    if (pid_setmov < -3.6)pid_setmov += 0.025;
break;
  case 6:  //range 60-70
    if (pid_setmov > -4.5)pid_setmov -= 0.025;
    if (pid_setmov < -4.5)pid_setmov += 0.025;
break;
  case 7:  //range 70-80
    if (pid_setmov > -5.4)pid_setmov -= 0.025;
    if (pid_setmov < -5.4)pid_setmov += 0.025;
break;
  case 8:  //range 80-90
    if (pid_setmov > -6.2)pid_setmov -= 0.025;
    if (pid_setmov < -6.2)pid_setmov += 0.025;
break;
  case 9:  //range 90-100
    if (pid_setmov > -6.6)pid_setmov -= 0.025;
    if (pid_setmov < -6.6)pid_setmov += 0.025;
break;
  }

}
```

mpu6050sudut.h

```
#ifndef MPU6050_SUDUT_H_
#define MPU6050_SUDUT_H_
  float raw_aX,raw_aY,raw_aZ;
  float aX,aY,aZ;
  float aRoll;
```

```
  float aPitch;
  float raw_gX,raw_gY,raw_gZ;
  float gX,gY,gZ;
  float gRoll;
  float gPitch;
  float dt;
  float Roll;
  float Pitch;
  int gyro_pitch_data_raw, gyro_yaw_data_raw,
accelerometer_data_raw;
  float angle_gyro, angle_acc, angle, angle_gyroraw,
alphagyro_raw;
  long gyro_x_cal_value;
  long gyro_y_cal_value;
  long gyro_yaw_calibration_value, gyro_pitch_calibration_value;
  int acc_calibration_value = -7782;

#endif
```

mpu6050sudut.ino

```
#include "Joystick.h"
#include "MotorStep.h"
#include "PID.h"
#include "SerialMon.h"
#include "mpu6050sudut.h"
#include "Strains.h"
#include "Arduino.h"
#include "HX711.h"

void mpu6050()
{
/*=====Setiing Sudut berdasarkan Accelorometer======*/
  Wire.beginTransmission(0x68); // Memulai Transmisi dari MPU6050
  Wire.write(0x3B);             // Menulis/mengambil data dari
Register Accelorometer
  Wire.endTransmission();   // mengakhiri transmisi
  while(Wire.available()<6) // Ketika data Byte yang tersedia < 6
  Wire.requestFrom(0x68,6);     // Mengambil 6 data Byte dimulai
dari 0x3B, 0x3C, 0x3D, 0x3E, 0x3F, 0x40
  raw_aX = Wire.read()<<8 | Wire.read(); // Membaca Nilai X_ACC
mentah/raw
```

```
raw_aY = Wire.read()<<8 | Wire.read(); // Membaca Nilai Y_ACC
mentah/raw
  raw_aZ = Wire.read()<<8 | Wire.read(); // Membaca Nilai Z_ACC
mentah/raw
  // Karena Menggunakan Full Scale Range +- 2g, maka data akan
dibagi dengan 16384 sesuai tabel pada datasheet.
  aX = raw_aX/16384;
  aY = raw_aY/16384;
  aZ = raw_aZ/16384;
  aPitch = -(atan2(aX,sqrt(aY*aY+aZ*aZ)))*57.2957795; // Nilai
Pitch diambil dari sensor Accelorometer
  aRoll = (atan2(aY,sqrt(aX*aX+aZ*aZ)))*57.2957795;   // Nilai
Roll diambil dari sensor Accelorometer

  /*=====Setiing Sudut berdasarkan Gyro======*/
  Wire.beginTransmission(0x68);
  //Start communication with the gyro
  Wire.write(0x43);          //Start reading at register 43
  Wire.endTransmission();    //End the transmission
  Wire.requestFrom(0x68, 4); //Request 4 bytes from the gyro
  gyro_yaw_data_raw = Wire.read()<<8|Wire.read();  //Combine the
two bytes to make one integer
  gyro_pitch_data_raw = Wire.read()<<8|Wire.read(); //Combine the
two bytes to make one integer

  gyro_pitch_data_raw -= gyro_y_cal_value;    //Add the gyro
calibration value
  //gyro_pitch_data_raw = gyro_pitch_data_raw/131;

  angle_gyroraw += gyro_pitch_data_raw*0.000031;
  angle_gyro += gyro_pitch_data_raw*0.000031;

  //bt = micros()-wt;
  //gyro_yaw_data_raw -= gyro_x_cal_value;
  //alphagyro_raw = aPitch + angle_gyroraw*bt;
  angle_gyroraw = angle_gyroraw * 1 + aPitch * 0;
  angle_gyro = angle_gyro * 0.99 + aPitch * 0.01;

}
```

B. Program Remote Controller

RemoteController.ino

```
#include <SPI.h>
#include <RF24.h>
#include <RF24Network.h>

RF24 radio(9,10);
RF24Network network(radio);

const uint16_t master00 = 00;
const uint16_t node01 = 01; //this node
//const uint16_t node02 = 02;


char data;

//#define JOYSTICK_X A2
//#define JOYSTICK_Y A3
//#define JOYSTICK_K 3
int joystick[2];
int key_pressed;
byte serial_output_joystick;
byte output_joystick;
byte gerak;
unsigned long wtr;
float incomingData;
int gerakin;

void setup()
{
  Serial.begin(9600);
  SPI.begin();
  radio.begin();
  network.begin(90, node01);
  radio.setDataRate(RF24_2MBPS);
  OCR2A = 39;
  wtr = millis()+500;

}
```

```
void loop()
{
  joystick[0] = analogRead(A2);//x
  joystick[1] = analogRead(A3);//y

  joystick[0] -= 512;
  joystick[1] = 512 - joystick[1];

  float alpha = atan2(joystick[1], joystick[0]);
  float power = sqrt(pow(joystick[0], 2) + pow(joystick[1], 2));
  //serial_output_joystick = Bluetooth1.write(gerak);
  if (power > 50)
  {
    //kanan
    if (alpha > -PI / 4*1.1 && alpha < PI / 4)
      gerak=B00000011;

    //belakang
    else if (alpha >= PI / 4 && alpha <=PI / 4 * 3)
      gerak=B00000010;

    //depan
    else if (alpha >= -PI / 4 * 3 && alpha <=-PI / 4*1.1)
      gerak=B00000001;

    //kiri
    else if (alpha < PI && alpha> -PI)
      gerak=B00000100;

    else
      gerak=B00000100;


  }//delay(20);
  else
  {
    gerak=B00000000;

  }
  //delay(5);
  //Serial.println(joystick[0]);
  //Serial.println(joystick[1]);
  //Serial.println(alpha);
  //Serial.println(gerak);
```

```
  gerakin=gerak;
  network.update();
  //int angka=8;
  radio.stopListening();
  RF24NetworkHeader header1(master00);
  bool ok1 = network.write(header1, &gerakin, sizeof(gerakin));

  radio.startListening();
 // delay(5);
   while (network.available()>0)
  {
   RF24NetworkHeader header1(master00);

    network.read(header1, &incomingData, sizeof(incomingData));
    Serial.println(incomingData);
  }


  while(wtr > millis());
  wtr += 500;


}
```

C. Program Pengolah Beban

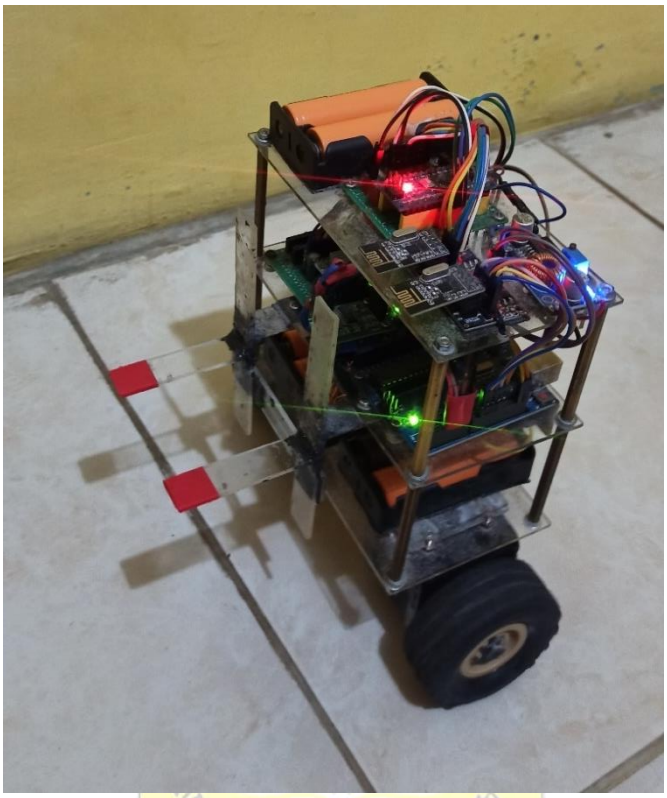
PengolahBeban.ino

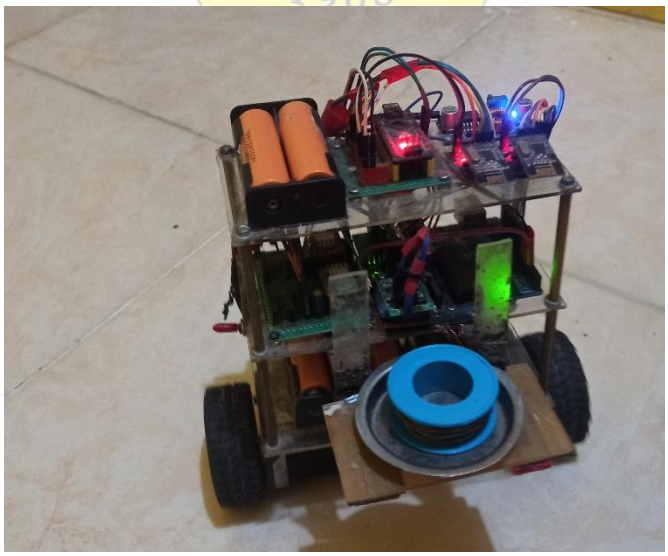
```
#include <RF24.h>
#include <RF24Network.h>
#include <SPI.h>
#include "HX711.h"
#include <Wire.h>
#define DOUT 6
#define CLK 7

RF24 radio(9,10);
RF24Network network(radio);
HX711 scale; //HX711 scale(6, 5);

const uint16_t this_node = 02;
//const uint16_t node01 = 01;
const uint16_t master00 = 00;
```

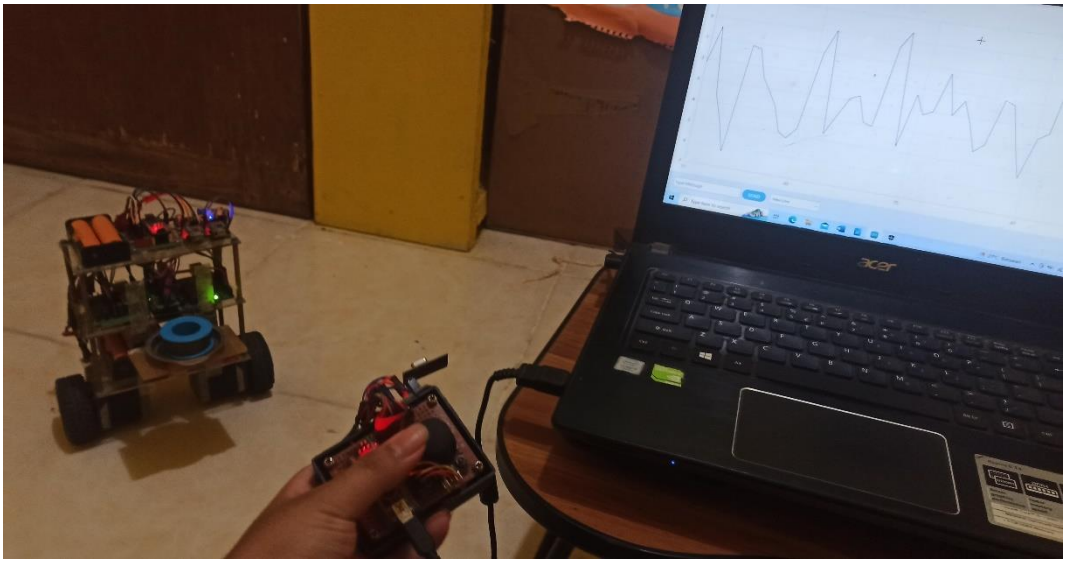
```
float calibration_factor = 6800;
float units;
int unit;
float ounces;
int sudut;
unsigned long wtr;
int coba;
int kene;
int test1;
int last_units;
unsigned long wtrq;
float gra;
float beban;

void setup()
{
  Serial.begin(9600);
  scale.begin(DOUT,CLK);
  SPI.begin();
  radio.begin();
  network.begin(90,this_node);
  radio.setDataRate(RF24_2MBPS);
  OCR2A = 39;
  Serial.println("HX711 weighing");
  scale.set_scale(calibration_factor);
  scale.tare(3);
  scale.read_average(1);
  scale.set_gain(128);
  scale.get_value(20);
  Serial.println("Readings:");
  //delay (20000);
  wtrq = millis()+500;
}

void loop()
{
  //delay(5);
  network.update();
  Serial.print("Reading:");
  units = scale.get_units(),1;
  beban = units*1.2246;
  unit = beban;
  ounces = units * 0.035274;
```

```
  Serial.print(unit);
  Serial.println(" grams");


  // int angka=99;
  RF24NetworkHeader header1(master00);
  bool ok1 = network.write(header1, &unit, sizeof(unit));

  last_units = test1;

  while(wtrq > millis());
  wtrq += 500;

}
```

Lampiran 2. Gambar Robot






acer